Windows Programming

Visual C++ MFC Programming

Lecture 10

김예진
Dept. of Game Software

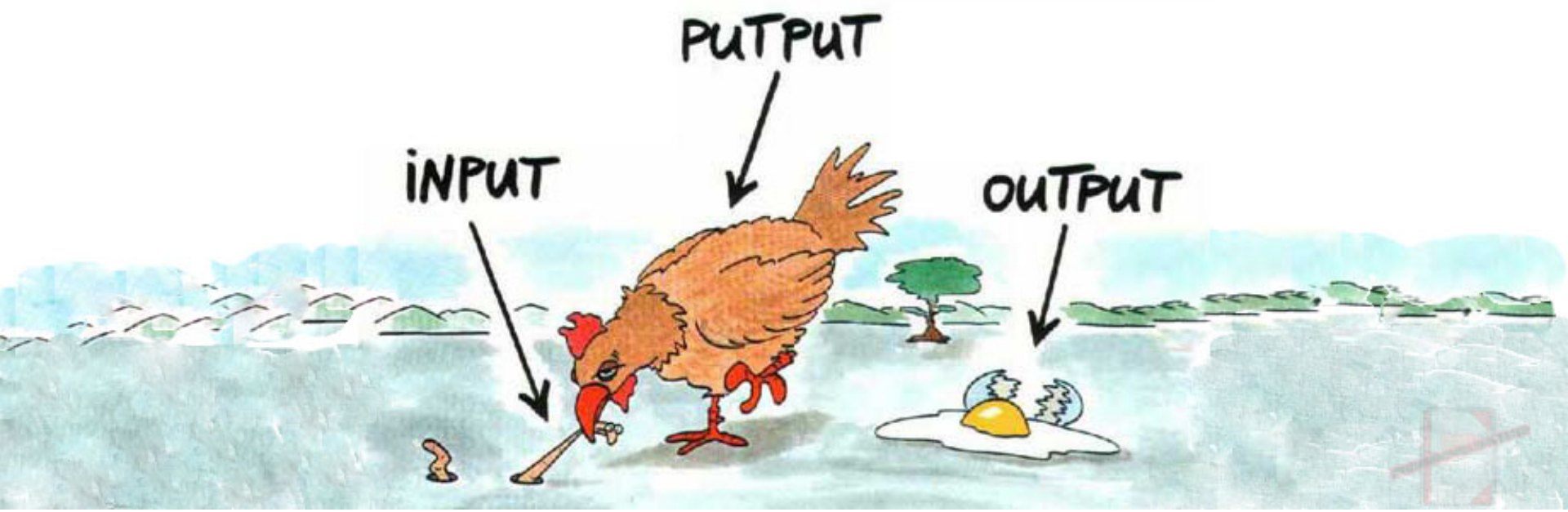
Notices

- 03/07: 502 → 501 등록 이동
- 03/21: HW 1 (Due: 03/28)
- 04/09: HW 2 (Due: 04/16)
- 04/25: Midterm (실습, ~75 min, 강의록 1~7)
- 05/14: HW 3 (Due: 05/21)

Plan

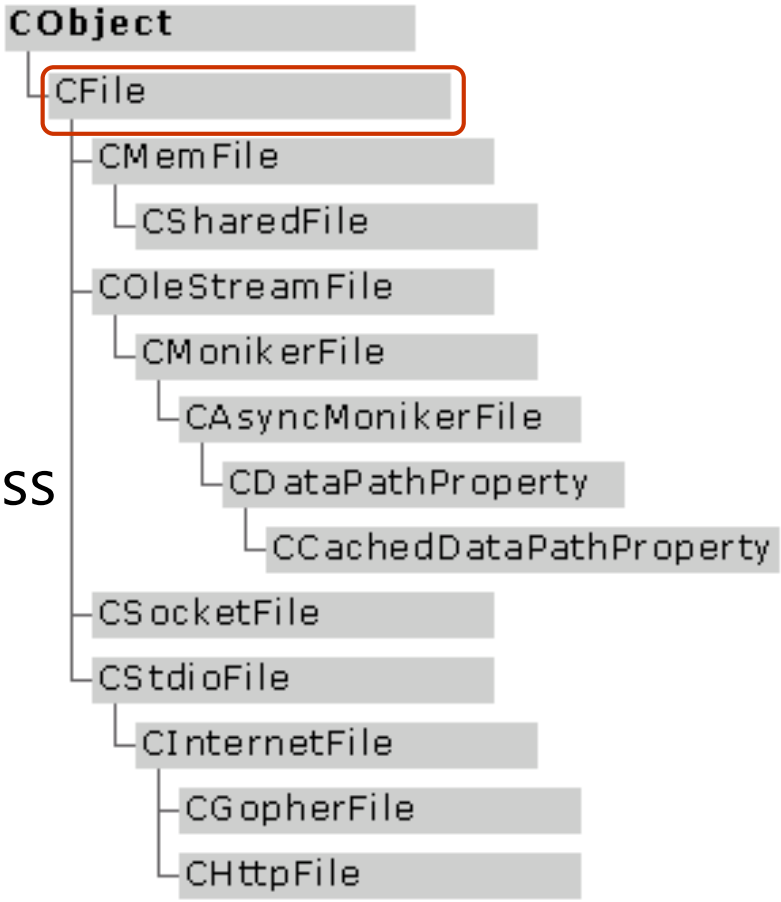
- File 입출력(Input & Output)
  - CFile class
  - 직렬화(Serialization)

File IO:
CFile Class



File 입출력

- 일반 file 입출력(IO): CFile class
  - File 입출력 기능 제공
  - 파생 class에 공통의 interface 제공
  - Read(), Write() 등의 함수 이용
  - 범용, 저수준(low-level)의 file IO
- 직렬화(serialization): CArchive class
  - << 또는 >> 연산자 이용
  - 편리한 이용
  - 사용 용도의 제한
  - 고수준(high-level)의 file IO



CFile

- 핵심 입출력 연산 과정
  1. File을 열거나 생성: Open( )
  2. File pointer의 위치에서 data를 읽음: Read( )
  3. File pointer의 위치에 data를 씀: Write( )
  4. File pointer의 위치를 변경: Seek( )
  5. File을 닫음: Close( )

- CFile로 file 열기와 생성
  - 방법 1: CFile 생성자 이용
  - 방법 2: CFile::Open member 함수 이용

CFile

- CFile 생성자를 이용한 file 열기와 생성

```
CFile file ( filename, mode );
```

  - File 접근 및 공유 mode

플래그	의미
CFile::modeCreate	File을 무조건 생성함 같은 이름의 file이 있다면 크기를 0으로 변경함
CFile::modeRead	읽기 전용 mode으로 file을 열거나 생성함
CFile::modeReadWrite	읽기 및 쓰기 mode로 file을 열거나 생성함
CFile::modeWrite	쓰기 전용 mode로 file을 열거나 생성함

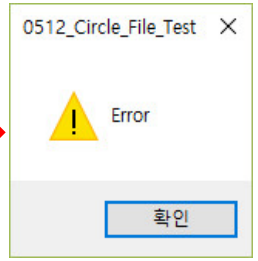
```
try
{
    CFile file(_T("mytest.txt"), CFile::modeReadWrite);
}

catch (CFileException* e)
{
    e->ReportError();
    e->Delete();
}
```

CFile

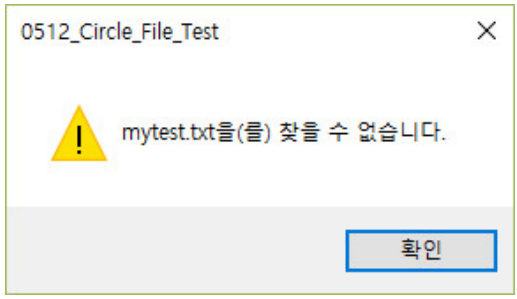
- CFile::Open member 함수를 이용한 파일 열기와 생성

```
CFile file;
file.Open( filename, mode, error );
```

```
CFile file;

if (file.Open(_T("mytest.txt"), CFile::modeRead) == false)
     AfxMessageBox(_T("Error"));
```




```
CFile file;
CFileException e;

if (!file.Open(_T("mytest.txt"), CFile::modeReadWrite, &e))
    e.ReportError();
```



CFile

- File 닫기
  - 소멸자에서 자동으로 닫아줌
  - 한 객체로 여러 file을 다룰 때는 Close member 함수 이용

```
CFile file;
CFileException e;

if (!file.Open(_T("mytest.txt"), CFile::modeReadWrite | CFile::modeCreate, &e))
{
      e.ReportError();
      return;
}

file.Close(); // 없으면 CFile::~CFile() 함수가 자동 호출됨
```

CFile

- File 읽기와 쓰기

```
UINT CFile::Read (void* lpBuf, UINT nCount);
void CFile::Write (const void* lpBuf, UINT nCount);
```

- File pointer 위치 변경

```
ULONGLONG CFile::Seek (LONGLONG lOff, UINT nFrom);
```

nFrom	의미
CFile::begin	File의 처음 위치부터 lOff만큼 file pointer 이동
CFile::current	현재의 file pointer 위치부터 lOff만큼 file pointer 이동
CFile::end	File의 끝 위치부터 lOff만큼 file pointer 이동

CFile

- 변수 저장

```
void CFile::Write (const void* lpBuf, UINT nCount) ;
```

```
CFile file(_T("test.txt"), CFile::modeCreate | CFile::modeWrite);

int a = 30, b = 20;
file.Write(&a, sizeof(a));
file.Write(&b, sizeof(b));

file.Close();
```

- 변수 읽기

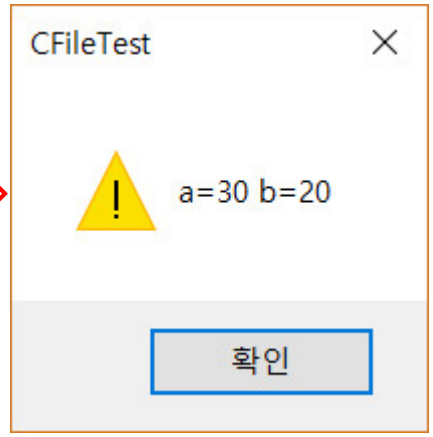
```
UINT CFile::Read (void* lpBuf, UINT nCount);
```

```
CFileException e;

if (!file.Open(_T("test.txt"), CFile::modeRead, &e)) {
      e.ReportError();
      return;
};

int a2 = 0, b2 = 0;
file.Read(&a2, sizeof(a2));
file.Read(&b2, sizeof(b2));

file.Close();
```

CFile

- 읽은 변수 출력

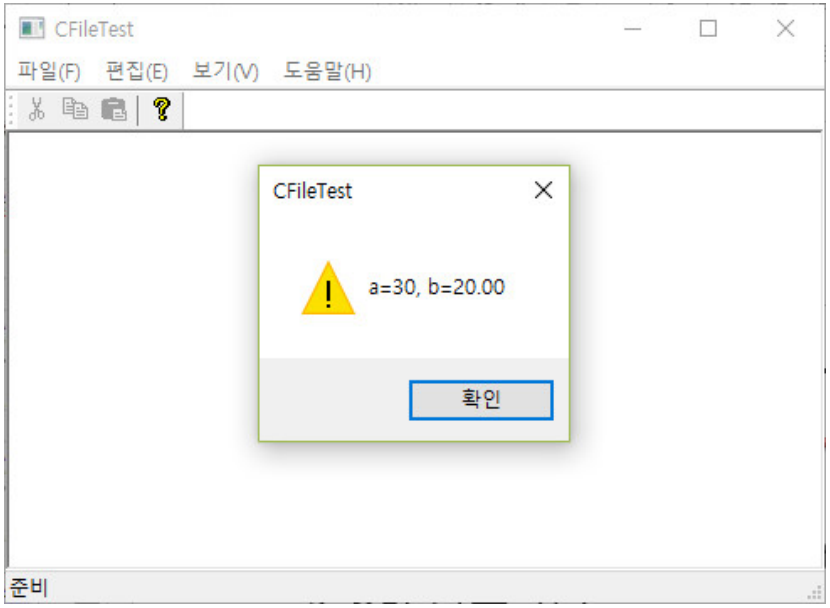
```
CString str;
str.Format(_T("a=%d b=%d"), a2, b2);
AfxMessageBox(str);
```



- 기타 함수
  - CFile::GetLength( ), CFile::SetLength( )
    - 파일의 현재 크기를 얻거나 변경
  - CFile::GetPosition( )
    - 현재의 파일 포인터 위치를 얻음
  - CFile::LockRange( ), CFile::UnlockRange( )
    - 파일의 일정 영역을 잠그거나 해제
    - 잠근 영역은 다른 process가 접근 못함
  - CFile::GetFilePath( ), CFile::GetFileName( )
    - 파일의 전체 경로(full path)와 이름을 얻음

CFile

- 사용 예: CFile로 file 저장 및 로딩
  1. 변수 int a, float b를 만듬
  2. LMB click으로 “test.dat”를 생성하고, a=30, b=20.0을 기록
  3. RMB click으로 “test.dat”을 열어 저장된 값을 읽고, a, b에 저장
  4. AfxMessageBox 함수를 이용하여 a, b 값을 출력



기타 File 관련 Class

- CMemFile: Memory 속에 가상의 file을 만들어 줌

```
CMemFile file;

// Memory file에 쓰기
int a = 100;
file.Write(&a, sizeof(a));

// Memory file에서 읽기
file.SeekToBegin();
int b;
file.Read(&b, sizeof(b));
TRACE("b = %d\n", b);
```

기타 File 관련 Class

- CStdioFile: Text file을 읽거나 쓰기
  - 문자열 읽기: ReadString( )

```
CString str;
file.ReadString(str);
```

  - 문자열 쓰기: WriteString( )

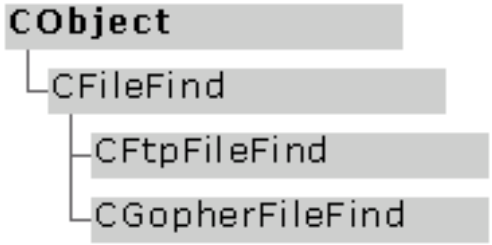
```
CString str =_T("Output");
file.WriteString(str);
```

- 사용 예: Text file에서 문자를 읽은 후 대문자로 바꿔 저장

```
CStdioFile file1;
file1.Open(_T("test1.txt"), CFile::modeRead);

CStdioFile file2;
file2.Open(_T("test2.txt"), CFile::modeWrite | CFile::modeCreate);

CString str;
while(file1.ReadString(str)) {
      str.MakeUpper();
      file2.WriteString(str + "\n");
}
```

기타 File 관련 Class

- CFileFind: Local disk에 대한 file 검색

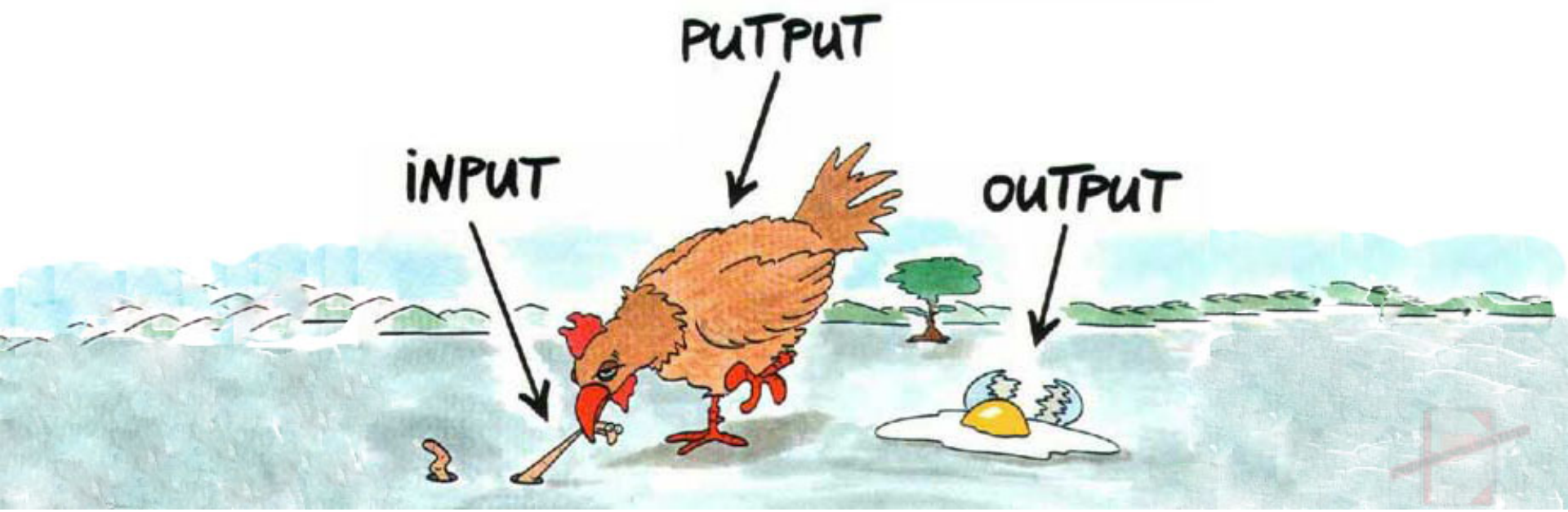



```
CFileFind finder;
bool bExist = finder.FindFile("MyText.txt");
```

- 사용 예: 현재 directory의 모든 file을 표시

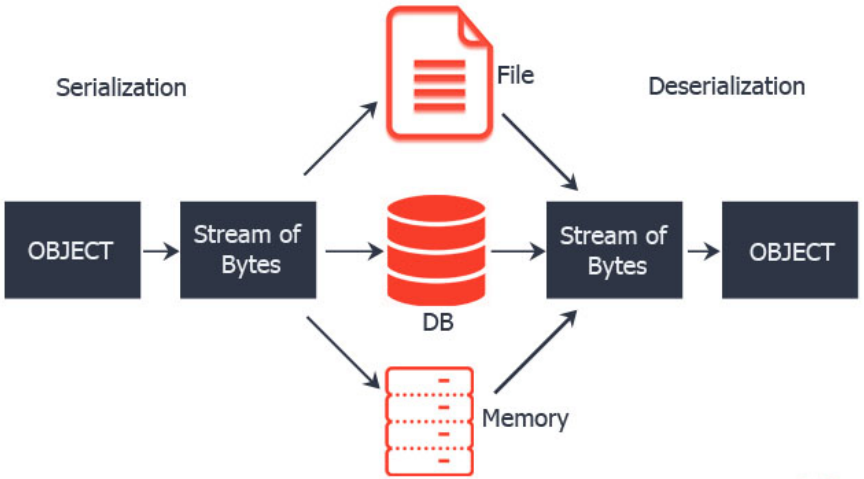
```
CFileFind finder;
BOOL bWorking = finder.FindFile("*.*");

while(bWorking) {
	bWorking = finder.FindNextFile();

	if(finder.IsDirectory())
		TRACE("[%s]\n", (LPCTSTR)finder.GetFileName());
	else
		TRACE("%s\n", (LPCTSTR)finder.GetFileName());
}
```

File IO:
Serialization



Serialization

- 직렬화(serialization)
  - 영속적인(persistitve) 저장 매체(e.g. disk file)에 object의 내용을 저장하거나 읽어오는 기법
  - **serialization** is … (*Wikipedia*)
    - *the process of **saving an object** onto a storage medium (such as a file, or a memory buffer) **to transmit** it across a network connection link in binary form.*
    - *The **series of bytes** or the format can be used to re-create an object that is identical in its internal state to the original object (actually, a clone).”*



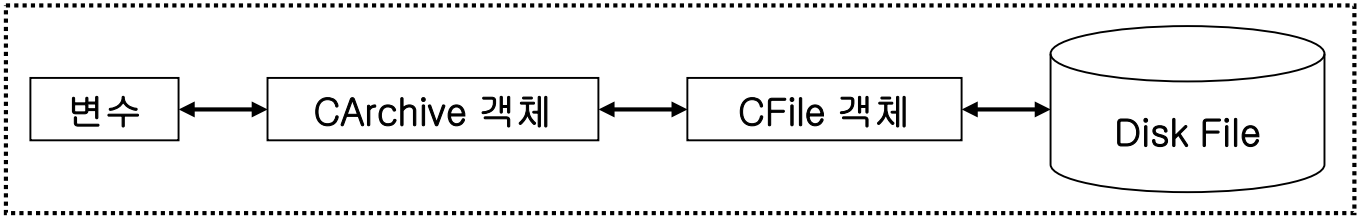
Serialization

- CArchive 생성자

```
CArchive::CArchive (CFile* pFile, UINT nMode, int nBufSize = 4096, void* lpBuf = NULL);
```

  - pFile: CFile 객체의 주소
  - nMode: CArchive::load 또는 CArchive::store
  - nBufSize: 내부에서 사용할 buffer 크기 (초기값 = 4096)
  - lpBuf: 사용자 정의 buffer의 주소 (초기값 = NULL)

- 직렬화 원리



Serialization

- CArchive data 쓰기
  - 일반 file 입출력 → 직렬화

```
CFile file;
CFileException e;

if(!file.Open(_T("mytest.dat"),
      CFile::modeReadWrite |
      CFile::modeCreate, &e))
{
    e.ReportError();
    return
}

int a = 30;
float b = 20.0f;

file.Write(&a, sizeof(a));
file.Write(&b, sizeof(b));

AfxMessageBox(_T("File Saved..!"));
```

```
CFile file;
CFileException e;

if(!file.Open(_T("mytest.dat"),
      CFile::modeReadWrite |
      CFile::modeCreate, &e))
{
    e.ReportError();
    return;
}

int a = 30;
float b = 20.0f;

CArchive ar (&file, CArchive::store);
ar << a << b;

AfxMessageBox(_T("File Saved..!"));
```

Serialization

- CArchive data 읽기
  - 일반 file 입출력 → 직렬화

```
CFile file;
CFileException e;

if(!file.Open(_T("mytest.dat"),
      CFile::modeRead, &e))
{
    e.ReportError();
    return;
}

int a = 0;
float b = 0;

file.Read(&a, sizeof(a));
file.Read(&b, sizeof(b));

CString str;
str.Format(_T("a=%d, b=%.2f"), a, b);
AfxMessageBox(str);
```

```
CFile file;
CFileException e;

if(!file.Open(_T("mytest.dat"),
      CFile::modeRead, &e))
{
    e.ReportError();
    return;
}

int a = 0;
float b = 0;

CArchive ar (&file, CArchive::load);
ar >> a >> b;

CString str;
str.Format(_T("a=%d, b=%.2f"), a, b);
AfxMessageBox(str);
```

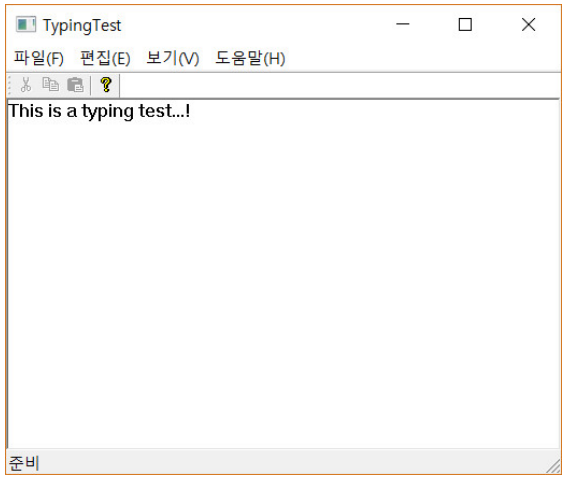
Serialization

- 직렬화 가능한 data type

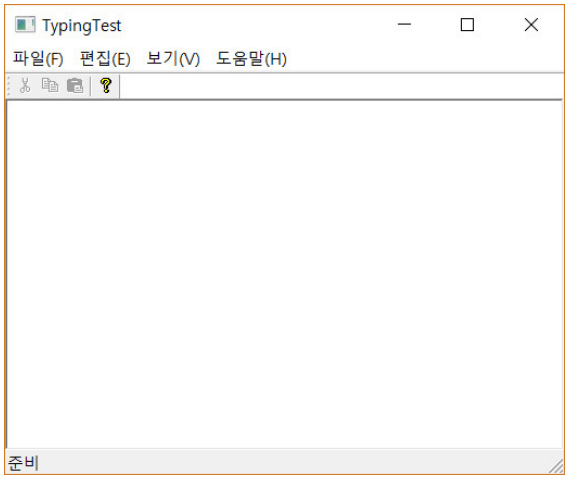
구분	Data Type
기본형	BYTE, WORD, LONG, DWORD, float, double, int, short, char, wchar_t, unsigned, bool, ULONGLONG, LONGLONG
비 기본형	RECT, POINT, SIZE, CRect, CPoint, CSize, CString, CTime, CTimeSpan, COleVariant, COleCurrency, COleDateTime, COleDataTimeSpan

Serialization

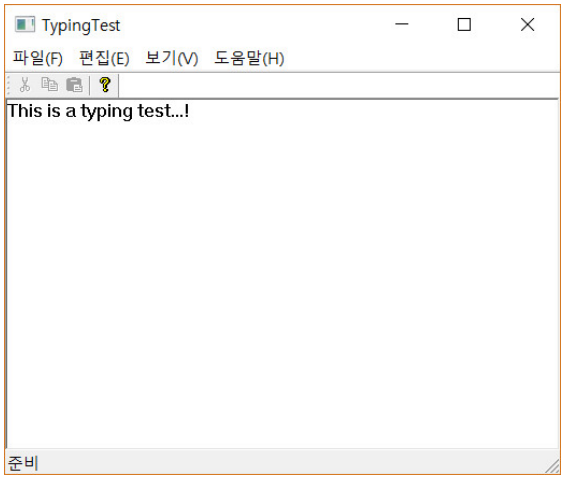
- 사용 예: Type한 문자 저장 및 로딩
  - Key를 입력 받아 사용자 영역에 출력
  - LMB: 출력 내용을 “testString.dat”에 저장
  - RMB: “testString.dat” 내용을 화면에 출력



Typing한 문자열 저장



프로그램 재시작



저장한 문자열 loading

Serialization

- 사용자 정의 class의 직렬화?

```
// Class 선언부
class CMyData
{
      public:
          CString m_str;
          COLORREF m_color;
      public:
          CMyData(CString &str, COLORREF &color) {
              m_str = str; m_color = color;
          }
          virtual ~CMyData();
};
```

```
// 직렬화 사용시
void CChildView::SaveOrLoad(CArchive& ar)
{
    if (ar.IsStoring()) {
        ar << m_data; // CMyData m_data;
    } else {
        ar >> m_data;
    }
}
```

➔ 사용자 정의 class에 적합한 <<, >> 연산자가 정의 안되어 있어 동작 안함
또한, standard 형식이 아님

Serialization

- 사용자 정의 class의 직렬화?
  - Microsoft 의 standard에 따라서 가능하게 변경

```
// Class 선언부
class CMyData : public CObject ①
{
    DECLARE_SERIAL(CMyData) ②
public:
    CString m_str;
    COLORREF m_color;
public:
    CMyData() { } ③
    CMyData(CString &str, COLORREF &color) {
        m_str = str; m_color = color;
    }
    void Serialize(CArchive& ar); ④
};

// Class 구현부
IMPLEMENT_SERIAL(CMyData, CObject, 1) ⑤

void CMyData::Serialize(CArchive& ar) ⑥
{
    CObject::Serialize(ar);
    if(ar.IsStoring())
        ar << m_str << m_color;
    else
        ar >> m_str >> m_color;
}
```

```
// 직렬화 사용시
void CChildView::SaveOrLoad(CArchive& ar)
{
    if (ar.IsStoring())
    {
        m_data.Serialize(ar); // CMyData m_data;
    }
    else
    {
        m_data.Serialize(ar);
    }
}
```

➔ Serialize 라는 함수를 호출하여 직렬화 수행: Standard에 따름

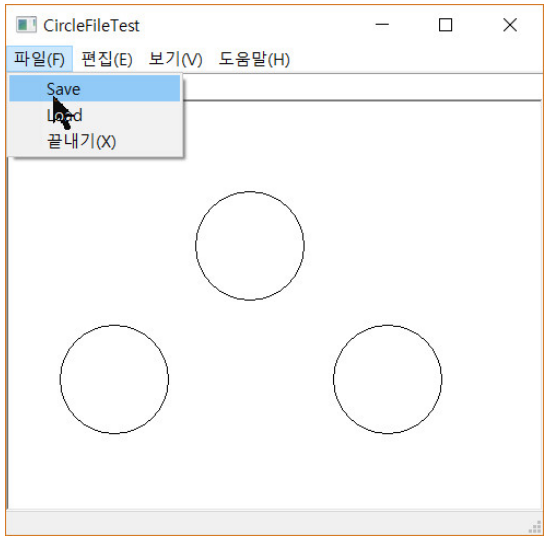
Serialization

- 직렬화의 단점
  - Data를 불러올 때 항상 file 전체를 읽음
  - Data를 저장할 때 항상 file 전체를 씀
  - Data 이외의 정보가 기록됨

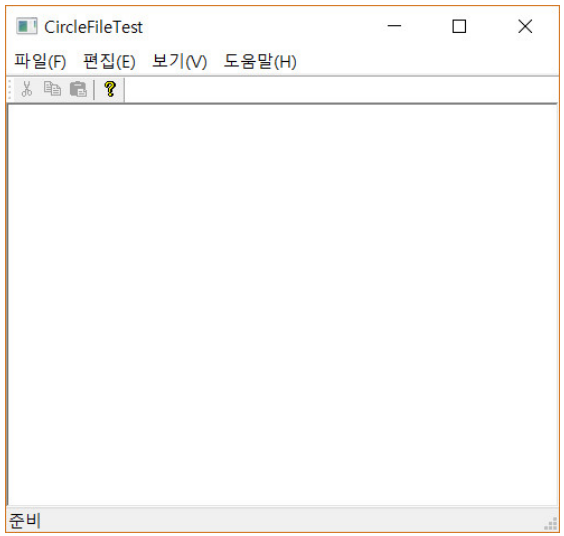
- 파일의 일부 내용만 읽어서 수정 후 다시 저장할 경우?
  프로그래머가 원하는 부분만 저장할 경우?
  → CFile class를 직접 사용하여 입출력 부분을 작성

Serialization

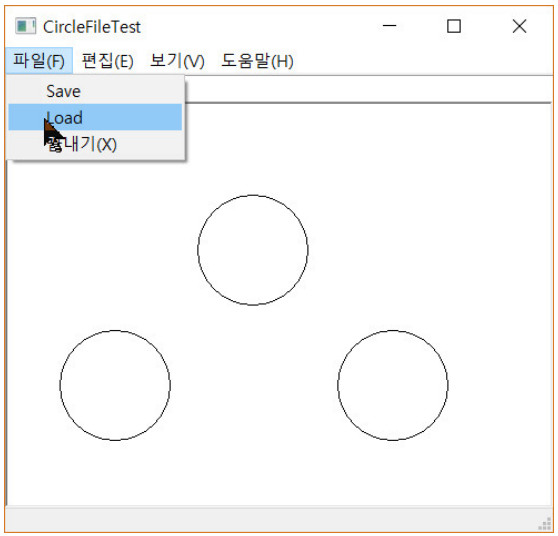
- 사용 예: 원 저장 및 로딩
  - LMB: 일정한 크기의 원을 다수 그림
  - Menu에 “Save”를 추가하고, 선택시 “circle.dat”에 그린 원들의 정보(위치)를 저장함
  - Menu에 “Load”를 추가하고, 선택시 “circle.dat”에서 원들의 정보를 복원한 후 화면에 그림



그린 원 저장



프로그램 재시작



저장한 원 loading

Q & A